MAIN FRAME PLUS★WARE™ NEWS
A PUBLICATION OF STSC, INC.
August 1986 Issue 5

In Decision by UK Petroleum Firm

APL★PLUS System Beats APL2

by Dave Mercer

In early 1985 a large UK petroleum company decided to migrate from VS APL. The choice was between IBM's APL2 and STSC's mainframe APL★PLUS® System. One very important application was running under APL—a large relational database. The system was the subject of continued development to enhance the facilities available to the users.

There were two main reasons for moving from VS APL: to allow the use of nested arrays in future development and to support multi-user access to the database.

After examining several alternatives the company decided that only two implementations could meet both requirements: Release 5 of STSC's APL★PLUS, and Version 1 of IBM's APL2 with STSC's SHAREFILE™AP. Both of these options offered nested array extensions, and a true shared file system; the user decided to evaluate both products.

The company needed to determine the effect that each product would have on the relational database in the areas of performance, relevance of new features, and potential improvements in speed of development.

The first step was to convert the database to run in both environments. The APL★PLUS conversion was carried out first. No changes were necessary, but several functions were modified to take advantage of APL★PLUS features. Where possible, utility functions were copied in from FASTFNS (a workspace supplied with APL★PLUS containing compiled functions) to replace existing VS APL code.

Several changes were required to the database before it could run under APL2, due to small modifications made in APL2 to several primitive functions (e.g., Format).

To measure performance, the original VS APL version and the two newly converted systems were timed while performing a set of typical database operations. APL★PLUS proved to be 20% faster than VS APL, while APL2 was 20% slower. Also, there appeared to be more scope for speeding up APL★PLUS than APL2 (particularly with the recent release of the compiler!) On investigation of the reasons for the speed discrepancy, it was found that look-up operations (dyadic iota) on numerics were up to 20 times faster using APL★PLUS. Other less dramatic differences

continued on page 4

Product Announcements

SQL Interface for APL★PLUS

Now users of the APL★PLUS System can access IBM's SQL/DS Relational Database Management System. STSC's new auxiliary processor APL★PLUS SQL Interface is modeled after the APL2/SQL auxiliary processor and performs any SQL/DS operation allowed under APL2.

The APL★PLUS SQL Interface is currently being reviewed by customers, and will be available this fall. It will be fully compatible with the APL2/SQL interface and will include a workspace with cover functions for easy interaction with SQL/DS.

Bring IBM's most powerful DBMS into your APL environment.

For more information about these new STSC products, call Jerry Turner, National Sales Manager, at (301) 984-5112.

Improved Formatting for APL2

The popular ⎕*FMT* enhanced formatting system function built into all APL★PLUS Systems is now available for APL2. The *FMT* function has been implemented as an APL2 External Function executed via Associated Processor 11. The basic behavior of the *FMT* External Function is identical to the ⎕*FMT* system function.

FMT provides a simple, economical and adaptable tool for detailed tabular formatting. Unlike the dyadic format primitive function, with *FMT* it is possible to format several data arguments at a time, to format data with decorations and special effects, to vary the printing order of columns of data from the normal left-to-right order, to insert message text, and to specify patterns for the data to follow.

The capabilities of *FMT* go far beyond the limited formatting available with APL2's "format by example". As a result, your reports look better and are easier to construct.

Efficient Character Searching

by Jim Weigang

APL's built-in searching capabilities are rather primitive. Often, programs need to be able to specify the search direction or the starting point of the search. Customarily, such variations are defined using array operations, but these techniques are needlessly wasteful. Functions in STSC's FASTFNS workspace provide efficient means of searching text strings for characters.

Dyadic Iota Variants

APL provides a single primitive function for searching, namely dyadic iota ($A\iota B$). Often, however, dyadic iota is not exactly what is needed; it may be necessary to start the search from somewhere other than the first element, or it may be necessary to search in reverse order, from last element to first. Usually, such variations of dyadic iota are implemented using dyadic iota in conjunction with drop ($A\downarrow B$) and/or reversal (ϕA). However, the array operations make these techniques needlessly inefficient. For example, consider the statement:

```
Z←I+(I↓TEXT)⍳CHAR
```

(which searches for *CHAR* in *TEXT* starting past the *I*th element). When executing this statement, the interpreter actually creates a temporary array for *I↓TEXT*, copying all but the first *I* elements of *TEXT* to the array. This effort is unnecessary since the search could simply be started with the $(I+1)$th element of *TEXT*. When a statement like the one above is imbedded in a loop, the result may be a critically inefficient program that executes in time proportional to $(\rho TEXT)*2$. By avoiding the array copying, the proportionality can be reduced to $(\rho TEXT)$.

These functions are written in assembly language and run almost as fast as if they were primitive operations built into the APL★PLUS System.

An Efficient Alternative

The *NEXTIN* function provides an efficient alternative to the drop-iota technique. *NEXTIN* searches for one of a set of characters starting at a specified location in a text vector. The text is stored in global variable *RINP*. The left argument to *NEXTIN* is the index of the first character in *RINP* to examine; the right argument is the characters to search for. For scalar *CHAR*, the statement above is equivalent to:

```
RINP←TEXT
Z←(I+1)NEXTIN CHAR
```

NEXTIN is more efficient because it does not make a copy of the vector, it just starts the search at the specified position. (Incidentally, on the Mainframe APL★PLUS System, *RINP←TEXT* passes a pointer, and does not actually make a copy of *TEXT*.) For short vectors, the extra overhead of calling a subroutine makes *NEXTIN* slightly slower, but for long vectors, *NEXTIN* can be much faster than drop-iota.

One application of *NEXTIN* is finding quoted strings in text. For example, the object might be to produce a Boolean mask with 1's marking quoted strings:

```
Input:   'IT''S' THIS 'AND' THAT
Output:  1111111000000111110000
```

A simple APL expression suffices to mark quoted strings if only one kind of quote character is allowed:

```
(≠\Q)∨Q←INPUT=''''
```

However, if the string may contain both single and double quotes, as in

```
"IT'S" THIS 'AND' THAT
```

the only known solution is to find the quoted strings one at a time in a loop. On each iteration, the program searches for quotes past some location in the text. Using drop and dyadic iota, the program runs in CPU time proportional to N-squared, where N is the length of the vector; using *NEXTIN*, the program runs in time proportional to N. For an 8000-element vector, the *NEXTIN* solution is 6.5 times faster than the drop-iota technique.

Other Useful FASTFNS

A related problem is to find the first occurence of a character *before* some point in a text vector. This is needed, for example, when adjusting text to fit within a specified width. The point to break a long line can be found by going out the maximum line length, then searching backwards for a blank or hyphen. The *PREVIN* function is like *NEXTIN*, but searches in the opposite direction. The expression

```
(POS+72) PREVIN ' -'
```

can be used to find a suitable break point for the line starting at *RINP*[*POS*], given a maximum length of 72. By using *PREVIN* for character searching, one customer speeded up a text width adjustment program by a factor of 6.

Another related problem is to perform a complementary search, looking for any character *not* occuring in a given list. The *PREVININ* function is similar to *PREVIN*, but searches for any character not in the right argument. One possible use of *PREVNIN* is to delete trailing blanks in a vector. The vector is scanned backwards from the end searching for a non-blank:

```
DTB:((⍴RINP)PREVNIN' ')↑RINP←⍵
```

For a 1000-element vector, this technique is 5.3 times faster than the conventional idiom $(\phi\vee\backslash\phi\omega\neq' ')/\omega$.

Finally, the *NEXTNIN* function is similar to *NEXTIN*, but searches for any character not in the right argument. *NEXTNIN* can be used to delete leading blanks in a vector with:

```
DLB:(¯1+1 NEXTNIN' ')↓RINP←⍵
```

Where Are They?

The *NEXTIN*, *NEXTNIN*, *PREVIN*, and *PREVNIN* functions can be found in the FASTFNS workspace in library 3. Online documentation is available in variable *NEXTIN DOC* in the same workspace. These functions are written in assembly language and run almost as fast as if they were primitive operations built into the APL★PLUS System.

Meet the Team

In this issue of *Mainframe PLUS★WARE News* we will introduce one of STSC's most distinguished senior systems developers, Robert Beilstein.

Bob joined the company in 1972 to help set up and run STSC's first independent computer center in Bethesda, Maryland. Since then he has designed and maintained the mainframe APL★PLUS System under the DOS, MVT, VM, and MVS operating systems. He is the author of STSC's FILESORT product, co-author of the SHAREFILE System, and wrote much of the APL★PLUS Compiler Support Package.

Bob has been working inside various APL systems since 1970. While pursuing an electrical engineering degree at Syracuse University, Bob attended a computing course taught by Thomas Pritchard, who is now also a senior systems programmer with STSC. It was the inspiration Bob needed to get involved with the Syracuse data center and turn to a career in computer science.

Bob's introduction to computers had come in 1965 as a radar technician in the Aleutian Islands of Alaska. There he was able to borrow time on the computer systems operating the radar site. Later he programmed and maintained a shipboard minicomputer system while in the Navy.

Robert J. Beilstein

Now Bob is married and still resides in Syracuse. His recent accomplishments include a trip through Europe trouble-shooting at STSC's international customer sites. He plans to continue working to improve the SHAREFILE System for Release 6 of the APL★PLUS System.

Bob Beilstein's vast systems experience and limitless creative energy are essential components of STSC's technical leadership in APL.

New APL Training Offered by STSC

STSC's Educational Services staff has announced new course offerings, an expanded system administrator's class, and a two-day educational conference at Princeton University.

System administrators may now enroll in an extended two-day class to learn practical techniques for APL★PLUS System operation and maintenance. This valuable workshop has been expanded to two days at the suggestion of earlier participants in order to allow attendees to share common experiences and concerns with other APL★PLUS System administrators. The second day of the System Administrators class is available to those who attended the old one-day class for a reduced fee.

The new course topics are "APL★PLUS Enhancements", "Mathematical Applications of APL", and "Pocket APL in One Day". Courses are available in both New York City and STSC headquarters in Rockville, Maryland.

STSC and the Princeton University School of Engineering will co-sponsor the October 29-30th conference, titled "Graphics and APL". Conference participants will spend two days learning how to use graphics and APL to enhance their power and productivity in using statistics. Neil Polhemus, of Statistical Graphics Corporation, will be a featured speaker. He will conduct a workshop that will focus on questions about using STATGRAPHICS.

For information about STSC's exciting lineup of classes, see the schedule below, or contact Tama Traberman, Educational Services Manager, at (212) 984-0730.

SCHEDULE OF APL CLASSES - 1986

Oct. 16-17	System Administrators	New York City
Oct. 29-30	Graphics and APL Confererence	Princeton, NJ
Oct. 20-23	APL for System Programmers	Rockville, MD
Nov. 17-20	APL for System Programmers	New York City
Dec. 1-4	Introductory APL	New York City
Dec. 8-11	Introductory APL	Rockville, MD
Aug. 18-21	Intermediate APL	Rockville, MD
Sep. 8-11	Intermediate APL	New York City
Sep. 22-25	APL★PLUS Enhancements	Rockville, MD
Oct. 6-9	APL★PLUS Enhancements	Rockville, MD
Oct. 28-31	Structured Programming in APL	Rockville, MD
Nov. 10-13	Structured Programming in APL	New York City
Nov. 10-13	Mathematical Applications of APL	Rockville, MD
Sep. 16	Pocket APL in One Day	New York City
Oct. 15	Pocket APL in One Day	New York City
Dec. 8	Pocket APL in One Day	New York City

MAIN FRAME PLUS★WARE NEWS
A PUBLICATION OF STSC, INC

Mainframe PLUS★WARE News is published by STSC, Inc. and is distributed free of charge to the company's customers. Comments regarding the newsletter should be sent to:

Christopher Bigler, Editor
STSC, Inc.
2115 East Jefferson St.
Rockville, MD 20852

©1986 STSC, Inc. All rights reserved. Printed in U.S.A.

APL★PLUS is a registered trademark of STSC, Inc.
PLUS★WARE and SHAREFILE are trademarks of STSC, Inc.

APL★PLUS Beats APL2 *continued from page 1*

were found, such as the sorting, which was marginally faster on APL★PLUS.

To determine the usefulness of the extended features offered by both new systems, minor developments were carried out using new facilities.

The first problem which became apparent with APL2 was EDITOR2 which is a definite retrograde step in relation to XEDIT. However, user-defined operators, $\square NA$ (when available) and alpha sort are definite improvements. APL★PLUS featured a host of new quad functions of which FI, VI, SS and FMT are particularly useful. The event control within APL★PLUS is far superior to APL2. The usefulness in terms of the application of features unique to APL2 such as complex numbers appeared limited, whereas APL★PLUS quad functions such as string search are extremely useful.

Only limited testing was performed on SHAREFILE as it was present in both systems. However, it is more closely linked to APL★PLUS as it forms an integral part of the interpreter, thus avoiding the overhead of an auxiliary processor such as is used by SHAREFILE AP, making the APL★PLUS implementation marginally faster and more efficient in use of memory. Also, APL★PLUS stores workspaces within SHAREFILE which is a feature not currently available for APL2. These differences were not critical to the company's application.

The technical information gained from the study was invaluable to the client, who subsequently decided to purchase APL★PLUS.

Dave Mercer is a senior consultant at Cocking and Drury, a London-based software consulting firm.

APL87 Call for Papers

The International Conference on APL, APL87, will be held May 11 to May 15, 1987 in Dallas, Texas. The APL87 program committee is soliciting abstracts for papers to address the issues of "APL in Transition".

The conference will have three main themes:

- APL as an Application Development System—Addressing such issues as user interface, transportability, ease of development, and maintenance and efficiency in operation.
- APL and its Environment—Papers on the interaction of APL and its host operating systems, other languages and packages. Also, papers on using APL in distributed processing or multiple CPU (or mainframe-PC) applications, and using APL with non-APL file systems.
- Future Developments in APL—including ISO standards, enhancements to the language, nested arrays, maintaining APL's technical superiority over alternate languages. Also including papers relating to strategies for increasing the use of APL, particularly among beginning users.

While these topics are only guidelines, the Program Committee encourages and will consider papers dealing with any subject relating to APL. The success of this conference depends on the participation of the entire APL community.

Informal abstracts are due to the Program Chairman by September 15, 1986. Completed papers are due by November 1, 1986, and are to be in English. Final copies of accepted papers are due by April 20, 1987, and must be suitable for photographic reproduction.

Program Chairman: Frank Cavallito
1812 Sherrye
Plano, Texas 75074
U.S.A.

Enhance Your APL Productivity With APL Press Publications

APL★PLUS® System users come in all sizes, shapes, and levels of experience. So whether you're a veteran or new APL user, you can enhance your APL productivity and skills with publications from STSC's APL Press book publishing division.

APL Press features a wide variety of reference manuals, textbooks, and tutorials. These include materials that not only teach APL programming techniques but also offer insight on how to use APL in a broad range of disciplines—science, engineering, probability theory, mathematics, and physics, just to name a few.

Some of the APL Press publications include:

- *APL Self-Study Course*, an interactive course in APL programming, complete with textbook, reference manual, and software, by Ken Iverson, the "father" of the APL language.
- *A Source Book in APL*, which provides background material on the fundamentals of the language.
- *Probability In APL*, a book/software combination that uses APL to present mathematical aspects of probability theory.
- *APL Quote-Quad: The Early Years*, an interesting and practical compilation of articles, programming notes, functions, APL algorithms, and technical papers documenting the development of APL technology from 1968 to 1972.
- *Collected Whizbangs, Vol. 1 and 2*, two collections of APL programming techniques and tricks that illustrate the powerful features of the language.

Take a look at the enclosed APL Press catalog for a complete listing and description of all APL Press publications. Feel free to make copies of the catalog to pass on to other APL users in your company.

To order APL Press publications, send in the attached order form. To order by phone or to get more copies of the catalog, call toll-free (800) 592-0050. In Maryland or Canada, call (301) 984-5123.

Register Now For New APL Courses! See Page 3.